Beyond Code Review: Detecting Errors via Context of Code Creation

Dušan ZELENÍK*

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova, 842 16 Bratislava, Slovakia
zelenik@fiit.stuba.sk

Every human has his own patterns in behavior. Everything we do has its reason. We react to different situations, different states. Therefore has the environment, in which we exist, huge impact on our outputs. When we talk about outputs, we have to mention our productivity and efficiency in work. It is definitely affected by our current situation and surroundings. Every human who is trying to accomplish something has to be focused on the task. Usually only experts repeating the routines do not need to focus so much that they are able to work in any state of their environment as it was discussed by Milton [1].In our work we focus on software developers and their productivity and effectiveness which is influenced by surroundings. Programmers do lot of mistakes when they are not in the shape. We are going to prove this assumption by analyzing their behavior while developing software. In our work we focus on two main aspects which affect the quality of programmer's outputs.

- *Continuity of work*. This means that programmer is working in continuous time and he is not interrupted by external happening. Interruptions could cause problems with refocusing on the task and reconstructing the situation. This leads to loosing time and loosing context and eventually leads to mistakes and incomplete tasks.
- *Stereotyped work*. When programmer works in common situations or in the common environment, he is used to conditions what positively affects his outputs. This also means that programmer needs some sort of stereotype in work. Loosing the stereotype brings anomalies in his behavior that causes anomalies in his outputs. Anomalies in outputs could emerge into mistakes.

Measuring the quality of programmer's outputs is not trivial issue. There are many different metrics which we could use. However most of them are not very precise and mostly incomparable among different programmers. Simple metric would be calculating the number of lines of operations in code which was created during specific

* Supervisor: Mária Bieliková, Institute of Informatics and Software Engineering

Spring 2013 PeWe Workshop, April 5, 2013, pp. 109–110.

amount of time. However, this only calculates the quantity but not quality of outputs. We assume that we should use two metrics.

- *Commits.* Commit is an action of programmer which is done regularly to submit some part of the work. Commits submitted into source control system are treated as logical end of programmer's effort. Commits are usually executed when something is accomplished. We consider number of commit as metric which shows the success.
- *Bugs.* Even if programmer has done something, we could cause some mistake which has, however, been revealed later. We are counting commits which are fixing bug associated with previous commits. Such a bugfix commits are negative metric which enables us to express low quality of programmer's work.

We are going to support the process of code reviewing by identifying part of codes which were created in bad conditions and are probable to contain some mistakes. Our approach lays in determining the context of programmer which could be associated with lower quality of outputs. We analyze programmer's activity by tracking his behavior. Next step is to detect parts of code which were created in this context.

In the work by Czerwinski et al. [2] authors analyzed behavior of workers. The nature of tasks of these workers was rather parallel. Multitasking suggests that these workers cannot work continuously on single task. They switch among tasks. This leads to frequent interruptions. The point of their work is to show how workers react to these interruptions. They want to design software for task management which is fond of bad habits caused by interruptions. Workers observed in this study are programmers. They work in Matlab but their tasks are usually fragmented due to secondary activities such as reading emails or preparing presentations. The study also showed that only 18% are dedicated to projects and productive tasks. The rest of the activities are secondary. 7% of total tasks are those which were interrupted by other tasks. 40% of tasks were self initiated, what means that user was not interrupted by eexternal influence but on his own. Rest of the interruptions was external.

Our method is based on detecting interruptions and anomalies in programmer's coding manners. Knowing when the programmer was interrupted and when he was working in inappropriate states we mark code which was produced. We mark this code with probability calculating while discovering interruptions and anomalies.

Acknowledgement. This contribution is the partial result of the Research & Development Operational Programme for the project Research of methods for acquisition, analysis and personalized conveying of information and knowledge, ITMS 26240220039, co-funded by the ERDF.

References

[1] Milton, J., Solodkin, A., Hluštík, P., Small, S.L.: The mind of expert motor performance is cool and focused. NeuroImage, 2007, vol. 35, no. 2, pp. 804–813.
[2] Czerwinski, M., Horvitz, E., Wilhite, S.: A diary study of task switching and interruptions. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '04, New York, NY, USA, ACM, 2004, pp. 175–182.